Network Working Group
Request for Comments: 5433
Category: Standards Track

T. Clancy
LTS
H. Tschofenig
Nokia Siemens Networks
February 2009

Extensible Authentication Protocol - Generalized Pre-Shared Key (EAP-GPSK) Method

Status of This Memo

Copyright Notice

Abstract

This memo defines an Extensible Authentication Protocol (EAP) method called EAP Generalized Pre-Shared Key (EAP-GPSK). This method is a lightweight shared-key authentication protocol supporting mutual authentication and key derivation.

Table of Contents

1. Introduction

EAP Generalized Pre-Shared Key (EAP-GPSK) is an EAP method defining a generalized pre-shared key authentication technique. Mutual authentication is achieved through a nonce-based exchange that is secured by a pre-shared key.

EAP-GPSK addresses a large number of design goals with the intention of being applicable in a broad range of usage scenarios.

The main design goals of EAP-GPSK are:

Simplicity:

EAP-GPSK should be easy to implement.

Security Model:

EAP-GPSK has been designed in a threat model where the attacker has full control over the communication channel. This EAP threat model is presented in Section 7.1 of [RFC3748].

Efficiency:

EAP-GPSK does not make use of public key cryptography and fully relies of symmetric cryptography. The restriction of symmetric cryptographic computations allows for low computational overhead. Hence, EAP-GPSK is lightweight and well suited for any type of device, especially those with processing power, memory, and battery constraints. Additionally, it seeks to minimize the number of round trips.

Flexibility:

EAP-GPSK offers cryptographic flexibility. At the beginning, the EAP server proposes a list of ciphersuites. The client then selects one. The current version of EAP-GPSK includes two ciphersuites, but additional ones can be easily added.

Extensibility:

The design of EAP-GPSK allows to securely exchange information between the EAP peer and the EAP server using protected data fields. These fields might, for example, be used to exchange channel binding information or to provide support for identity confidentiality.

2. Terminology

In this document, several words are used to signify the requirements of the specification. These words are often capitalized. The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

This section describes the various variables and functions used in the EAP-GPSK method.

Variables:

CSuite_List: An octet array listing available ciphersuites (variable length).

CSuite_Sel: Ciphersuite selected by the peer (6 octets).

ID_Peer: Peer Network Access Identifier (NAI) [RFC4282].

ID_Server: Server identity as an opaque blob.

KS: Integer representing the input key size, in octets, of the selected ciphersuite CSuite_Sel. The key size is one of the ciphersuite parameters.

ML: Integer representing the length of the Message Authentication Code (MAC) output, in octets, of the selected ciphersuite CSuite_Sel.

PD_Payload: Data carried within the protected data payload.

PD_Payload_Block: Block of possibly multiple PD_Payloads carried by a GPSK packet.

PL: Integer representing the length of the PSK in octets (2 octets). PL MUST be larger than or equal to KS.

RAND_Peer: Random integer generated by the peer (32 octets).

RAND_Server: Random integer generated by the server (32 octets).

Operations:

A || B: Concatenation of octet strings A and B.

A**B: Integer exponentiation.

truncate(A,B): Returns the first B octets of A.

ENC_X(Y): Encryption of message Y with a symmetric key X, using a defined block cipher.

KDF-X(Y): Key Derivation Function that generates an arbitrary number of octets of output using secret X and seed Y.

length(X): Function that returns the length of input X in octets, encoded as a 2-octet integer in network byte order.

MAC_X(Y): Keyed message authentication code computed over Y with symmetric key X.

SEC_X(Y): SEC is a function that provides integrity protection based on the chosen ciphersuite. The function SEC uses the algorithm defined by the selected ciphersuite and applies it to the message content Y with key X. In short, SEC_X(Y) = Y || MAC_X(Y).

X[A..B]: Notation representing octets A through B of octet array X where the first octet of the array has index zero.

The following abbreviations are used for the keying material:

EMSK: Extended Master Session Key is exported by the EAP method (64 octets).

MK: A session-specific Master Key between the peer and EAP server from which all other EAP method session keys are derived (KS octets).

MSK: Master Session Key exported by the EAP method (64 octets).

PK: Session key generated from the MK and used during protocol exchange to encrypt protected data (KS octets).

PSK: Long-term key shared between the peer and the server (PL octets).

SK: Session key generated from the MK and used during protocol exchange to demonstrate knowledge of the PSK (KS octets).

3. Overview

The EAP framework (see Section 1.3 of [RFC3748]) defines three basic steps that occur during the execution of an EAP conversation between the EAP peer, the Authenticator, and the EAP server.

1. The first phase, discovery, is handled by the underlying protocol, e.g., IEEE 802.1X as utilized by IEEE 802.11 [80211].

2. The EAP authentication phase with EAP-GPSK is defined in this document.

3. The secure association distribution and secure association phases are handled differently depending on the underlying protocol.

EAP-GPSK performs mutual authentication between the EAP peer ("Peer") and EAP server ("Server") based on a pre-shared key (PSK). The protocol consists of the message exchanges (GPSK-1, ..., GPSK-4) in which both sides exchange nonces and their identities, and compute and exchange a Message Authentication Code (MAC) over the previously exchanged values, keyed with the pre-shared key. This MAC is considered as proof of possession of the pre-shared key. Two further messages, namely GPSK-Fail and GPSK-Protected-Fail, are used to deal with error situations.

A successful protocol exchange is shown in Figure 1.

```
   +--------+                                     +--------+
   |        |                EAP-Request/Identity |        |
   |  EAP   |<------------------------------------|  EAP   |
   |  peer  |                                     | server |
   |        | EAP-Response/Identity               |        |
   |        |------------------------------------>|        |
   |        |                                     |        |
   |        |                  EAP-Request/GPSK-1 |        |
   |        |<------------------------------------|        |
   |        |                                     |        |
   |        | EAP-Response/GPSK-2                 |        |
   |        |------------------------------------>|        |
   |        |                                     |        |
   |        |                  EAP-Request/GPSK-3 |        |
   |        |<------------------------------------|        |
   |        |                                     |        |
   |        | EAP-Response/GPSK-4                 |        |
   |        |------------------------------------>|        |
   |        |                                     |        |
   |        |          EAP-Success                |        |
   |        |<------------------------------------|        |
   +--------+                                     +--------+
```

Figure 1: EAP-GPSK: Successful Exchange

The full EAP-GPSK protocol is as follows:

GPSK-1:

ID_Server, RAND_Server, CSuite_List

GPSK-2:

SEC_SK(ID_Peer, ID_Server, RAND_Peer, RAND_Server, CSuite_List, CSuite_Sel, [ENC_PK(PD_Payload_Block)])

GPSK-3:

SEC_SK(RAND_Peer, RAND_Server, ID_Server, CSuite_Sel, [ENC_PK(PD_Payload_Block)])

GPSK-4:

SEC_SK([ENC_PK(PD_Payload_Block)])

The EAP server begins EAP-GPSK by selecting a random number RAND_Server and encoding the supported ciphersuites into CSuite_List. A ciphersuite consists of an encryption algorithm, a key derivation function, and a message authentication code.

In GPSK-1, the EAP server sends its identity ID_Server, a random number RAND_Server, and a list of supported ciphersuites CSuite_List. The decision of which ciphersuite to offer and which ciphersuite to pick is policy- and implementation-dependent and, therefore, outside the scope of this document.

In GPSK-2, the peer sends its identity ID_Peer and a random number RAND_Peer. Furthermore, it repeats the received parameters of the GPSK-1 message (ID_Server, RAND_Server, CSuite_List) and the selected ciphersuite. It computes a Message Authentication Code over all the transmitted parameters.

The EAP server verifies the received Message Authentication Code and the consistency of the identities, nonces, and ciphersuite parameters transmitted in GPSK-1. In case of successful verification, the EAP server computes a Message Authentication Code over the session parameter and returns it to the peer (within GPSK-3). Within GPSK-2 and GPSK-3, the EAP peer and EAP server have the possibility to exchange encrypted protected data parameters.

The peer verifies the received Message Authentication Code and the consistency of the identities, nonces, and ciphersuite parameters transmitted in GPSK-2. If the verification is successful, GPSK-4 is prepared. This message can optionally contain the peer's protected data parameters.

Upon receipt of GPSK-4, the server processes any included PD_Payload_Block. Then, the EAP server sends an EAP Success message to indicate the successful outcome of the authentication.

4. Key Derivation

EAP-GPSK provides key derivation in compliance to the requirements of [RFC3748] and [RFC5247]. Note that this section provides an abstract description for the key derivation procedure that needs to be instantiated with a specific ciphersuite.

The long-term credential shared between EAP peer and EAP server SHOULD be a strong pre-shared key PSK of at least 16 octets, though its length and entropy are variable. While it is possible to use a password or passphrase, doing so is NOT RECOMMENDED as EAP-GPSK is vulnerable to dictionary attacks.

During an EAP-GPSK authentication, a Master Key MK, a Session Key SK, and a Protected Data Encryption Key PK (if using an encrypting ciphersuite) are derived using the ciphersuite-specified KDF and data exchanged during the execution of the protocol, namely 'RAND_Peer || ID_Peer || RAND_Server || ID_Server', referred to as inputString in its short-hand form.

In case of successful completion, EAP-GPSK derives and exports an MSK and an EMSK, each 64 octets in length.

The following notation is used: KDF-X(Y, Z)[A..B], whereby

X is the length, in octets, of the desired output,

Y is a secret key,

Z is the inputString,

[A..B] extracts the string of octets starting with octet A and finishing with octet B from the output of the KDF function.

This keying material is derived using the ciphersuite-specified KDF as follows:

- o inputString = RAND_Peer || ID_Peer || RAND_Server || ID_Server
- o MK = KDF-KS(PSK[0..KS-1], PL || PSK || CSuite_Sel || inputString)[0..KS-1]
- o MSK = KDF-{128+2*KS}(MK, inputString)[0..63]
- o EMSK = KDF-{128+2*KS}(MK, inputString)[64..127]
- o SK = KDF-{128+2*KS}(MK, inputString)[128..127+KS]
- o PK = KDF-{128+2*KS}(MK, inputString)[128+KS..127+2*KS] (if using an encrypting ciphersuite)

The value for PL (the length of the PSK in octets) is encoded as a 2-octet integer in network byte order. Recall that KS is the length of the ciphersuite input key size in octets.

Additionally, the EAP keying framework [RFC5247] requires the definition of a Method-ID, Session-ID, Peer-ID, and Server-ID. These values are defined as:

- o Method-ID = KDF-16(PSK[0..KS-1], "Method ID" || EAP_Method_Type || CSuite_Sel || inputString)[0..15]

o Session-ID = EAP_Method_Type || Method_ID

o Peer-ID = ID_Peer

o Server-ID = ID_Server

EAP_Method_Type refers to the 1-octet, IANA-allocated EAP Type code value.

Figure 2 depicts the key derivation procedure of EAP-GPSK.

```
+-------------+     +-------------------------------+
|   PL-octet  |     | RAND_Peer || ID_Peer ||       |
|     PSK     |     | RAND_Server || ID_Server      |
+-------------+     +-------------------------------+
       |                            |            |
       |     +------------+         |            |
       |     | CSuite_Sel |         |            |
       |     +------------+         |            |
       |           |                |            |
       v           v                v            |
+--------------------------------------------+   |
|                    KDF                     |   |
+--------------------------------------------+   |
                         |                       |
                         v                       |
                 +-------------+                 |
                 |   KS-octet  |                 |
                 |      MK     |                 |
                 +-------------+                 |
                         |                       |
                         v                       v
+---------------------------------------------------+
|                       KDF                         |
+---------------------------------------------------+
     |             |             |            |
     v             v             v            v
+---------+   +---------+  +----------+  +----------+
| 64-octet|   | 64-octet|  | KS-octet |  | KS-octet |
|   MSK   |   |  EMSK   |  |    SK    |  |    PK    |
+---------+   +---------+  +----------+  +----------+
```

Figure 2: EAP-GPSK Key Derivation

5. Key Management

In order to be interoperable, PSKs must be entered in the same way on both the peer and server. The management interface for entering PSKs MUST support entering PSKs up to 64 octets in length as ASCII strings and in hexadecimal encoding.

Additionally, the ID_Peer and ID_Server MUST be provisioned with the PSK. Validation of these values is by an octet-wise comparison. The management interface SHOULD support entering non-ASCII octets for the ID_Peer and ID_Server up to 254 octets in length. For more information, the reader is advised to read Section 2.4 of RFC 4282 [RFC4282].

6. Ciphersuites

The design of EAP-GPSK allows cryptographic algorithms and key sizes, called ciphersuites, to be negotiated during the protocol run. The ability to specify block-based and hash-based ciphersuites is offered. Extensibility is provided with the introduction of new ciphersuites; this document specifies an initial set. The CSuite/Specifier column in Figure 3 uniquely identifies a ciphersuite.

For a vendor-specific ciphersuite, the first four octets are the vendor-specific enterprise number that contains the IANA-assigned "SMI Network Management Private Enterprise Codes" value (see [ENTNUM]), encoded in network byte order. The last two octets are vendor assigned for the specific ciphersuite. A vendor code of 0x00000000 indicates ciphersuites standardized by the IETF in an IANA-maintained registry.

The following ciphersuites are specified in this document (recall that KS is the length of the ciphersuite input key length in octets, and ML is the length of the MAC output in octets):

CSuite/ Specifier	KS	Encryption	ML	Integrity / KDF MAC	Key Derivation Function
0x0001	16	AES-CBC-128	16	AES-CMAC-128	GKDF
0x0002	32	NULL	32	HMAC-SHA256	GKDF

Figure 3: Ciphersuites

Ciphersuite 1, which is based on the Advanced Encryption Standard (AES) as a cryptographic primitive, MUST be implemented. This document specifies also a second ciphersuite, which MAY be implemented. Both ciphersuites defined in this document make use of the Generalized Key Derivation Function (GKDF), as defined in Section 7. The following aspects need to be considered to ensure that the PSK that is used as input to the GKDF is sufficiently long:

1. The PSK used with ciphersuite 1 MUST be 128 bits in length. Keys longer than 128 bits will be truncated.

2. The PSK used with ciphersuite 2 MUST be 256 bits in length. Keys longer than 256 bits will be truncated.

3. It is RECOMMENDED that 256 bit keys be provisioned in all cases to provide enough entropy for all current and many possible future ciphersuites.

Ciphersuites defined in the future that make use of the GKDF need to specify a minimum PSK size (as is done with the ciphersuites listed in this document).

7. Generalized Key Derivation Function (GKDF)

Each ciphersuite needs to specify a key derivation function. The ciphersuites defined in this document make use of the Generalized Key Derivation Function (GKDF) that utilizes the MAC function defined in the ciphersuite. Future ciphersuites can use any other formally specified KDF that takes as arguments a key and a seed value, and produces at least 128+2*KS octets of output.

GKDF has the following structure:

GKDF-X(Y, Z)

X length, in octets, of the desired output

Y secret key

Z inputString

```
GKDF-X (Y, Z)
{
  n = ceiling integer of ( X / ML );
     /* determine number of output blocks */
```

```
     result = "";

     for i = 1 to n {
       result = result || MAC_Y (i || Z);
     }

     return truncate(result, X)
   }
```

Note that the variable 'i' in M_i is represented as a 2-octet value in network byte order.

8. Ciphersuites Processing Rules

8.1. Ciphersuite #1

8.1.1. Encryption

With this ciphersuite, all cryptography is built around a single cryptographic primitive, AES-128 ([AES]). Within the protected data frames, AES-128 is used in the Cipher Block Chaining (CBC) mode of operation (see [CBC]). This EAP method uses encryption in a single payload, in the protected data payload (see Section 9.4).

In a nutshell, the CBC mode proceeds as follows. The IV is XORed with the first plaintext block before it is encrypted. Then for successive blocks, the previous ciphertext block is XORed with the current plaintext, before it is encrypted.

8.1.2. Integrity

Ciphersuite 1 uses CMAC as Message Authentication Code. CMAC is recommended by NIST. Among its advantages, CMAC is capable to work with messages of arbitrary length. A detailed description of CMAC can be found in [CMAC].

The following instantiation is used: AES-CMAC-128(SK, Input) denotes the MAC of Input under the key SK where Input refers to the following content:

- o Parameter within SEC_SK(Parameter) in message GPSK-2
- o Parameter within SEC_SK(Parameter) in message GPSK-3
- o Parameter within SEC_SK(Parameter) in message GPSK-4

8.2. Ciphersuite #2

8.2.1. Encryption

Ciphersuite 2 does not include an algorithm for encryption. With a NULL encryption algorithm, encryption is defined as:

E_X(Y) = Y

When using this ciphersuite, the data exchanged inside the protected data block is not encrypted. Therefore, this mode MUST NOT be used if confidential information appears inside the protected data block.

8.2.2. Integrity

Ciphersuite 2 uses the keyed MAC function HMAC, with the SHA256 hash algorithm (see [RFC4634]).

For integrity protection, the following instantiation is used:

HMAC-SHA256(SK, Input) denotes the MAC of Input under the key SK where Input refers to the following content:

- Parameter within SEC_SK(Parameter) in message GPSK-2
- Parameter within SEC_SK(Parameter) in message GPSK-3
- Parameter within SEC_SK(Parameter) in message GPSK-4

9. Packet Formats

This section defines the packet format of the EAP-GPSK messages.

9.1. Header Format

The EAP-GPSK header has the following structure:

```
--- bit offset --->
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Code      |  Identifier   |            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |    OP-Code    |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
|                                                               |
...                         Payload                           ...
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: EAP-GPSK Header

The Code, Identifier, Length, and Type fields are all part of the EAP header and are defined in [RFC3748]. The Type field in the EAP header MUST be the value allocated by IANA for EAP-GPSK.

The OP-Code field is one of 6 values:

- 0x00 : Reserved
- 0x01 : GPSK-1
- 0x02 : GPSK-2
- 0x03 : GPSK-3
- 0x04 : GPSK-4
- 0x05 : GPSK-Fail
- 0x06 : GPSK-Protected-Fail

All other values of this OP-Code field are available via IANA registration.

9.2. Ciphersuite Formatting

Ciphersuites are encoded as 6-octet arrays. The first four octets indicate the CSuite/Vendor field. For vendor-specific ciphersuites, this represents the vendor enterprise number and contains the IANA-assigned "SMI Network Management Private Enterprise Codes" value (see [ENTNUM]), encoded in network byte order. The last two octets indicate the CSuite/Specifier field, which identifies the particular ciphersuite. The 4-octet CSuite/Vendor value 0x00000000 indicates ciphersuites allocated by the IETF.

Graphically, they are represented as:

```
--- bit offset --->
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       CSuite/Vendor = 0x00000000 or enterprise number         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      CSuite/Specifier         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5: Ciphersuite Formatting

CSuite_Sel is encoded as a 6-octet ciphersuite CSuite/Vendor and CSuite/Specifier pair.

CSuite_List is a variable-length octet array of ciphersuites. It is encoded by concatenating encoded ciphersuite values. Its length in octets MUST be a multiple of 6.

9.3. Payload Formatting

Payload formatting is based on the protocol exchange description in Section 3.

The GPSK-1 payload format is defined as follows:

```
--- bit offset --->
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       length(ID_Server)       |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
|                                                               |
...                         ID_Server                         ...
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
...                   32-octet RAND_Server                    ...
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      length(CSuite_List)      |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
|                                                               |
...                        CSuite_List                        ...
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 6: GPSK-1 Payload

The GPSK-2 payload format is defined as follows:

```
--- bit offset --->
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         length(ID_Peer)       |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
|                                                               |
...                         ID_Peer                           ...
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        length(ID_Server)      |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
|                                                               |
...                         ID_Server                         ...
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
...                     32-octet RAND_Peer                    ...
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
...                    32-octet RAND_Server                   ...
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       length(CSuite_List)     |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
|                                                               |
...                        CSuite_List                        ...
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            CSuite_Sel                         |
+                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |   length(PD_Payload_Block)    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
...                 optional PD_Payload_Block                 ...
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
...                   ML-octet payload MAC                    ...
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 7: GPSK-2 Payload

If the optional protected data payload is not included, then length(PD_Payload_Block)=0 and the PD payload is excluded. The payload MAC covers the entire packet, from the ID_Peer length through the optional PD_Payload_Block.

The GPSK-3 payload is defined as follows:

```
--- bit offset --->
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
...                    32-octet RAND_Peer                     ...
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
...                    32-octet RAND_Server                     ...
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       length(ID_Server)       |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
|                                                               |
...                         ID_Server                           ...
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            CSuite_Sel                         |
+                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |   length(PD_Payload_Block)    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
...                 optional PD_Payload_Block                   ...
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
...                   ML-octet payload MAC                      ...
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 8: GPSK-3 Payload

If the optional protected data payload is not included, then length(PD_Payload_Block)=0 and the PD payload is excluded. The payload MAC covers the entire packet, from the RAND_Peer through the optional PD_Payload_Block.

The GPSK-4 payload format is defined as follows:

```
--- bit offset --->
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   length(PD_Payload_Block)    |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
|                                                               |
...                 optional PD_Payload_Block                 ...
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
...                   ML-octet payload MAC                    ...
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 9: GPSK-4 Payload

If the optional protected data payload is not included, then length(PD_Payload_Block)=0 and the PD payload is excluded. The MAC MUST always be included, regardless of the presence of PD_Payload_Block. The payload MAC covers the entire packet, from the PD_Payload_Block length through the optional PD_Payload_Block.

The GPSK-Fail payload format is defined as follows:

```
--- bit offset --->
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Failure-Code                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 10: GPSK-Fail Payload

The GPSK-Protected-Fail payload format is defined as follows:

```
--- bit offset --->
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Failure-Code                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
...                   ML-octet payload MAC                    ...
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 11: GPSK-Protected-Fail Payload

The Failure-Code field is one of three values, but can be extended:

- o 0x00000000 : Reserved
- o 0x00000001 : PSK Not Found
- o 0x00000002 : Authentication Failure
- o 0x00000003 : Authorization Failure

All other values of this field are available via IANA registration.

"PSK Not Found" indicates a key for a particular user could not be located, making authentication impossible. "Authentication Failure" indicates a MAC failure due to a PSK mismatch. "Authorization Failure" indicates that while the PSK being used is correct, the user is not authorized to connect.

9.4. Protected Data

The protected data blocks are a generic mechanism for the peer and server to securely exchange data. If the specified ciphersuite has a NULL encryption primitive, then this channel only offers authenticity, not confidentiality.

These payloads are encoded as the concatenation of type-length-value (TLV) triples called PD_Payloads.

Type values are encoded as a 6-octet string and represented by a 4-octet vendor and a 2-octet specifier field. The vendor field indicates the type as either standards-specified or vendor-specific.

If these four octets are 0x00000000, then the value is standards-specified, and any other value represents a vendor-specific enterprise number.

The specifier field indicates the actual type. For vendor field 0x00000000, the specifier field is maintained by IANA. For any other vendor field, the specifier field is maintained by the vendor.

Length fields are specified as 2-octet integers in network byte order, reflect only the length of the value, and do not include the length of the type and length fields.

Graphically, this can be depicted as follows:

```
--- bit offset --->
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   PData/Vendor                                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
         PData/Specifier        |         PData/Length          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
...                          PData/Value                      ...
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 12: Protected Data Payload (PD_Payload) Formatting

These PD_Payloads are concatenated together to form a PD_Payload_Block. If the CSuite_Sel includes support for encryption, then the PD_Payload_Block includes fields specifying an Initialization Vector (IV) and the necessary padding. This can be depicted as follows:

```
   --- bit offset --->
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |   IV Length   |                                               |
   +-+-+-+-+-+-+-+-+      Initialization Vector                    +
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   ...                          PD_Payload                       ...
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   ...                  optional PD_Payload, etc                 ...
   |                                                               |
   +               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |               |             Padding (0-255 octets)            |
   +-+-+-+-+-+-+-+-+                               +-+-+-+-+-+-+-+-+
   |                                               |  Pad Length   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 13: Protected Data Block (PD_Payload_Block)
Formatting if Encryption is Supported

The Initialization Vector is a randomly chosen value whose length is equal to the specified IV Length. The required length is defined by the ciphersuite. Recipients MUST accept any value. Senders SHOULD either pick this value pseudo-randomly and independently for each message or use the final ciphertext block of the previous message sent. Senders MUST NOT use the same value for each message, use a sequence of values with low hamming distance (e.g., a sequence number), or use ciphertext from a received message. IVs should be selected per the security requirements of the underlying cipher. If the data is not being encrypted, then the IV Length MUST be 0. If the ciphersuite does not require an IV, or has a self-contained way of communicating the IV, then the IV Length field MUST be 0. In these cases, the ciphersuite definition defines how the IV is encapsulated in the PD_Payload.

The concatenation of PD_Payloads along with the padding and padding length are all encrypted using the negotiated block cipher. If no block cipher is specified, then these fields are not encrypted.

The Padding field MAY contain any value chosen by the sender. For block-based cipher modes, the padding MUST have a length that makes the combination of the concatenation of PD_Payloads, the Padding, and the Pad Length to be a multiple of the encryption block size. If the

underlying ciphersuite does not require padding (e.g., a stream-based cipher mode) or no encryption is being used, then the padding length MUST still be present and be 0.

The Pad Length field is the length of the Padding field. The sender SHOULD set the Pad Length to the minimum value that makes the combination of the PD_Payloads, the Padding, and the Pad Length a multiple of the block size (in the case of block-based cipher modes), but the recipient MUST accept any length that results in proper alignment. This field is encrypted with the negotiated cipher.

If the negotiated ciphersuite does not support encryption, then the IV field MUST be of length 0 and the padding field MUST be of length 0. The IV length and padding length fields MUST still be present, and contain the value 0. The rationale for still requiring the length fields is to allow for modular implementations where the crypto processing is independent of the payload processing. This is depicted in the following figure.

```
--- bit offset --->
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      0x00     |                                               |
+-+-+-+-+-+-+-+-+          PD_Payload                         ...
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
...              optional PD_Payload, etc    +-+-+-+-+-+-+-+-+
|                                               |      0x00     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 14: Protected Data Block (PD_Payload_Block)
Formatting Without Encryption

For PData/Vendor field 0x00000000, the following PData/Specifier fields are defined:

- o 0x0000 : Reserved

All other values of this field are available via IANA registration.

10. Packet Processing Rules

This section defines how the EAP peer and EAP server MUST behave when a received packet is deemed invalid.

Any EAP-GPSK packet that cannot be parsed by the EAP peer or the EAP server MUST be silently discarded. An EAP peer or EAP server receiving any unexpected packet (e.g., an EAP peer receiving GPSK-3 before receiving GPSK-1 or before transmitting GPSK-2) MUST silently discard the packet.

GPSK-1 contains no MAC protection, so provided it properly parses, it MUST be accepted by the peer. If the EAP peer has no ciphersuites in common with the server or decides the ID_Server is that of an Authentication, Authorization, and Accounting (AAA) server to which it does not wish to authenticate, the EAP peer MUST respond with an EAP-NAK.

For GPSK-2, if the ID_Peer is for an unknown user, the EAP server MUST send either a "PSK Not Found" GPSK-Fail message or an "Authentication Failure" GPSK-Fail, depending on its policy. If the MAC validation fails, the server MUST transmit a GPSK-Fail message specifying "Authentication Failure". If the RAND_Server or CSuite_List field in GPSK-2 does not match the values in GPSK-1, the server MUST silently discard the packet. If server policy determines the peer is not authorized and the MAC is correct, the server MUST transmit a GPSK-Protected-Fail message indicating "Authorization Failure", and discard the received packet.

A peer receiving a GPSK-Fail / GPSK-Protected-Fail message in response to a GPSK-2 message MUST replay the received GPSK-Fail / GPSK-Protected-Fail message. Then, the EAP server returns an EAP-Failure after receiving the GPSK-Fail / GPSK-Protected-Fail message to correctly finish the EAP conversation. If MAC validation on a GPSK-Protected-Fail packet fails, then the received packet MUST be silently discarded.

For GPSK-3, a peer MUST silently discard messages where the RAND_Peer, ID_Server, or the CSuite_Sel fields do not match those transmitted in GPSK-2. An EAP peer MUST silently discard any packet whose MAC fails.

For GPSK-4, a server MUST silently discard any packet whose MAC fails validation.

If a decryption failure of a protected payload is detected, the recipient MUST silently discard the GPSK packet.

11. Example Message Exchanges

This section shows a couple of example message flows.

A successful EAP-GPSK message exchange is shown in Figure 1.

```
   +--------+                                     +--------+
   |        |                EAP-Request/Identity |        |
   |  EAP   |<------------------------------------|  EAP   |
   |  peer  |                                     | server |
   |        | EAP-Response/Identity               |        |
   |        |------------------------------------>|        |
   |        |                                     |        |
   |        |                  EAP-Request/GPSK-1 |        |
   |        |<------------------------------------|        |
   |        |                                     |        |
   |        | EAP-Response/EAP-NAK                |        |
   |        |------------------------------------>|        |
   |        |                                     |        |
   |        |          EAP-Failure                |        |
   |        |<------------------------------------|        |
   +--------+                                     +--------+
```

Figure 15: EAP-GPSK: Unsuccessful Exchange
(Unacceptable AAA Server Identity; ID_Server)

```
   +--------+                                     +--------+
   |        |                EAP-Request/Identity |        |
   |  EAP   |<------------------------------------|  EAP   |
   |  peer  |                                     | server |
   |        | EAP-Response/Identity               |        |
   |        |------------------------------------>|        |
   |        |                                     |        |
   |        |                  EAP-Request/GPSK-1 |        |
   |        |<------------------------------------|        |
   |        |                                     |        |
   |        | EAP-Response/GPSK-2                 |        |
   |        |------------------------------------>|        |
   |        |                                     |        |
   |        | EAP-Request/GPSK-Fail               |        |
   |        | (PSK Not Found or Authentication    |        |
   |        | Failure)                            |        |
   |        |<------------------------------------|        |
   |        |                                     |        |
   |        | EAP-Response/GPSK-Fail              |        |
   |        | (PSK Not Found or Authentication    |        |
   |        | Failure)                            |        |
   |        |------------------------------------>|        |
   |        |                                     |        |
   |        |          EAP-Failure                |        |
   |        |<------------------------------------|        |
   +--------+                                     +--------+
```

Figure 16: EAP-GPSK: Unsuccessful Exchange (Unknown User)

```
   +--------+                                     +--------+
   |        |                EAP-Request/Identity |        |
   |  EAP   |<------------------------------------|  EAP   |
   |  peer  |                                     | server |
   |        | EAP-Response/Identity               |        |
   |        |------------------------------------>|        |
   |        |                                     |        |
   |        |                  EAP-Request/GPSK-1 |        |
   |        |<------------------------------------|        |
   |        |                                     |        |
   |        | EAP-Response/GPSK-2                 |        |
   |        |------------------------------------>|        |
   |        |                                     |        |
   |        | EAP-Request/GPSK-Fail               |        |
   |        | (Authentication Failure)            |        |
   |        |<------------------------------------|        |
   |        |                                     |        |
   |        | EAP-Response/GPSK-Fail              |        |
   |        | (Authentication Failure)            |        |
   |        |------------------------------------>|        |
   |        |                                     |        |
   |        |          EAP-Failure                |        |
   |        |<------------------------------------|        |
   +--------+                                     +--------+
```

Figure 17: EAP-GPSK: Unsuccessful Exchange (Invalid MAC in GPSK-2)

```
   +--------+                                     +--------+
   |        |                EAP-Request/Identity |        |
   |  EAP   |<------------------------------------|  EAP   |
   |  peer  |                                     | server |
   |        | EAP-Response/Identity               |        |
   |        |------------------------------------>|        |
   |        |                                     |        |
   |        |                  EAP-Request/GPSK-1 |        |
   |        |<------------------------------------|        |
   |        |                                     |        |
   |        | EAP-Response/GPSK-2                 |        |
   |        |------------------------------------>|        |
   |        |                                     |        |
   |        | EAP-Request/                        |        |
   |        | GPSK-Protected-Fail                 |        |
   |        | (Authorization Failure)             |        |
   |        |<------------------------------------|        |
   |        |                                     |        |
   |        | EAP-Request/                        |        |
   |        | GPSK-Protected-Fail                 |        |
   |        | (Authorization Failure)             |        |
   |        |------------------------------------>|        |
   |        |                                     |        |
   |        |          EAP-Failure                |        |
   |        |<------------------------------------|        |
   +--------+                                     +--------+
```

Figure 18: EAP-GPSK: Unsuccessful Exchange (Authorization Failure)

12. Security Considerations

[RFC3748] highlights several attacks that are possible against EAP since EAP itself does not provide any security.

This section discusses the claimed security properties of EAP-GPSK as well as vulnerabilities and security recommendations in the threat model of [RFC3748].

12.1. Security Claims

Authentication mechanism: Shared Keys
Ciphersuite negotiation: Yes (Section 12.16)
Mutual authentication: Yes (Section 12.2)
Integrity protection: Yes (Section 12.4)
Replay protection: Yes (Section 12.5)
Confidentiality: No (Section 12.17, Section 12.15)
Key derivation: Yes (Section 12.8)
Key strength: Varies (Section 12.8)

Dictionary attack protection: No (Section 12.7)
Fast reconnect: No (Section 12.14)
Cryptographic binding: N/A (Section 12.18)
Session independence: Yes (Section 12.10)
Fragmentation: No (Section 12.12)
Channel binding: Extensible (Section 12.13)

12.2. Mutual Authentication

EAP-GPSK provides mutual authentication.

The server believes that the peer is authentic when it successfully verifies the MAC in the GPSK-2 message; the peer believes that the server is authentic when it successfully verifies the MAC it receives with the GPSK-3 message.

The key used for mutual authentication is derived based on the long-term secret PSK, nonces contributed by both parties, and other parameters. The long-term secret PSK has to provide sufficient entropy and, therefore, sufficient strength. The nonces (RAND_Peer and RAND_Server) need to be fresh and unique for every session. In this way, EAP-GPSK is not different than other authentication protocols based on pre-shared keys.

12.3. Protected Result Indications

EAP-GPSK supports protected result indications via the GPSK-Protected-Fail message. This allows a server to provide additional information to the peer as to why the session failed, and to do so in an authenticated way (if possible). In particular, the server can indicate the lack of PSK (account not present), failed authentication (PSK incorrect), or authorization failure (account disabled or unauthorized). Only the third message could be integrity protected.

It should be noted that these options make debugging network and account errors easier, but they also leak information about accounts to attackers. An attacker can determine if a particular ID_Peer is a valid user on the network or not. Thus, implementers should use care in enabling this particular option on their servers. If they are in an environment where such attacks are of concern, then protected result indication capabilities should be disabled.

12.4. Integrity Protection

EAP-GPSK provides integrity protection based on the ciphersuites suggested in this document. Integrity protection is a minimum feature every ciphersuite must provide.

12.5. Replay Protection

EAP-GPSK provides replay protection of its mutual authentication part thanks to the use of random numbers RAND_Server and RAND_Peer. Since RAND_Server is 32 octets long, one expects to have to record 2**64 (i.e., approximately 1.84*10**19) EAP-GPSK successful authentications before a protocol run can be replayed. Hence, EAP-GPSK provides replay protection of its mutual authentication part as long as RAND_Server and RAND_Peer are chosen at random; randomness is critical for replay protection. RFC 4086 [RFC4086] describes techniques for producing random quantities.

12.6. Reflection Attacks

Reflection attacks occur in bi-directional, challenge-response, mutual authentication protocols where an attacker, upon being issued a challenge by an authenticator, responds by issuing the same challenge back to the authenticator, obtaining the response, and then "reflecting" that same response to the original challenge.

EAP-GPSK provides protection against reflection attacks because the message formats for the challenges differ. The protocol does not consist of two independent authentications, but rather the authentications are tightly coupled.

Also note that EAP-GPSK does not provide MAC protection of the OP-Code field, but again since each message is constructed differently, it would not be possible to change the OP-Code of a valid message and still have it be parseable and accepted by the recipient.

12.7. Dictionary Attacks

EAP-GPSK relies on a long-term shared secret (PSK) that SHOULD be based on at least 16 octets of entropy to be fully secure. The EAP-GPSK protocol makes no special provisions to ensure keys based on passwords are used securely. Users who use passwords as the basis of their PSK are not protected against dictionary attacks. Derivation of the long-term shared secret from a password is strongly discouraged.

The success of a dictionary attack against EAP-GPSK depends on the strength of the long-term shared secret (PSK) it uses. The PSK used by EAP-GPSK SHOULD be drawn from a pool of secrets that is at least 2^128 bits large and whose distribution is uniformly random. Note that this does not imply resistance to dictionary attacks -- only that the probability of success in such an attack is acceptably remote.

12.8. Key Derivation and Key Strength

EAP-GPSK supports key derivation as shown in Section 4.

Keys used within EAP-GPSK are all based on the security of the originating PSK. PSKs SHOULD have at least 16 octets of entropy. Independent of the protocol exchange (i.e., without knowing RAND_Peer and RAND_Server), the keys have been derived with sufficient input entropy to make them as secure as the underlying KDF output key length.

12.9. Denial-of-Service Resistance

There are three forms of denial-of-service (DoS) attacks relevant for this document, namely (1) attacks that lead to a vast amount of state being allocated, (2) attacks that attempt to prevent communication between the peer and server, and (3) attacks against computational resources.

In an EAP-GPSK conversation the server has to maintain state, namely the 32-octet RAND_Server, when transmitting the GPSK-1 message to the peer. An adversary could therefore flood a server with a large number of EAP-GPSK communication attempts. An EAP server may therefore ensure that an established state times out after a relatively short period of time when no further messages are received. This enables a sort of garbage collection.

The client has to keep state information after receiving the GPSK-1 message. To prevent a replay attack, all the client needs to do is ensure that the value of RAND_Peer is consistent between GPSK-2 and GPSK-3. Message GPSK-3 contains all the material required to re-compute the keying material. Thus, if a client chooses to implement this client-side DoS protection mechanism, it may manage RAND_Peer and CSuite_Sel on a per-server basis for servers it knows, instead of on a per-message basis.

Attacks that disrupt communication between the peer and server are mitigated by silently discarding messages with invalid MACs. Attacks against computational resources are mitigated by having very light-weight cryptographic operations required during each protocol round.

The security considerations of EAP itself, see Sections 5.2 and 7 of RFC 3748 [RFC3748], are also applicable to this specification (e.g., for example concerning EAP-based notifications).

12.10. Session Independence

Thanks to its key derivation mechanisms, EAP-GPSK provides session independence: passive attacks (such as capture of the EAP conversation) or active attacks (including compromise of the MSK or EMSK) do not enable compromise of subsequent or prior MSKs or EMSKs. The assumption that RAND_Peer and RAND_Server are random is central for the security of EAP-GPSK in general and session independence in particular.

12.11. Compromise of the PSK

EAP-GPSK does not provide perfect forward secrecy. Compromise of the PSK leads to compromise of recorded past sessions.

Compromise of the PSK enables the attacker to impersonate the peer and the server, and it allows the adversary to compromise future sessions.

EAP-GPSK provides no protection against a legitimate peer sharing its PSK with a third party. Such protection may be provided by appropriate repositories for the PSK, the choice of which is outside the scope of this document. The PSK used by EAP-GPSK must only be shared between two parties: the peer and the server. In particular, this PSK must not be shared by a group of peers (e.g., those with different ID_Peer values) communicating with the same server.

The PSK used by EAP-GPSK must be cryptographically separated from keys used by other protocols, otherwise the security of EAP-GPSK may be compromised.

12.12. Fragmentation

EAP-GPSK does not support fragmentation and reassembly since the message size is relatively small. However, it should be noted that this impacts the length of protected data payloads that can be attached to messages. Also, if the EAP frame is larger than the MTU of the underlying transport, and that transport does not support fragmentation, the frame will most likely not be transported. Consequently, implementers and deployers should take care to ensure EAP-GPSK frames are short enough to work properly on the target underlying transport mechanism.

12.13. Channel Binding

This document enables the ability to exchange channel binding information. It does not, however, define the encoding of channel binding information in the document.

12.14. Fast Reconnect

EAP-GPSK does not provide fast reconnect capability since this method is already at (or close to) the lower limit of the number of roundtrips and the cryptographic operations.

12.15. Identity Protection

Identity protection is not specified in this document. Extensions can be defined that enhance this protocol to provide this feature.

12.16. Protected Ciphersuite Negotiation

EAP-GPSK provides protected ciphersuite negotiation via the indication of available ciphersuites by the server in the first message, and a confirmation by the peer in the subsequent message.

Note, however, that the GPSK-2 message may optionally contain a payload, ENC_PK(PD_Payload_Block), protected with an algorithm based on a selected ciphersuite before the ciphersuite list has actually been authenticated. In the classical downgrading attack, an adversary would choose a ciphersuite that is so weak that it can be broken in real time or would attempt to disable cryptographic protection altogether. The latter is not possible since any ciphersuite defined for EAP-GPSK must at least provide authentication and integrity protection. Confidentiality protection is optional. When, at some time in the future, a ciphersuite contains algorithms that can be broken in real-time, then a policy on peers and the server needs to indicate that such a ciphersuite must not be selected by any of parties.

Furthermore, an adversary may modify the selection of the ciphersuite for the client to select a ciphersuite that does not provide confidentiality protection. As a result, this would cause the content of PD_Payload_Block to be transmitted in cleartext. When protocol designers extend EAP-GPSK to carry information in the PD_Payload_Block of the GPSK-2 message, then it must be indicated whether confidentiality protection is mandatory. In case such an extension requires a ciphersuite with confidentiality protection, then the policy at the peer must be to not transmit information of that extension in the PD_Payload_Block of the GPSK-2 message. The peer may, if possible, delay the transmission of this information element to the GPSK-4 message where the ciphersuite negotiation has been confirmed already. In general, when a ciphersuite is selected that does not provide confidentiality protection, then information that demands confidentiality protection must not be included in any of the PD_Payload_Block objects.

12.17. Confidentiality

Although EAP-GPSK provides confidentiality in its protected data payloads, it cannot claim to do so, per Section 7.2.1 of [RFC3748], since it does not support identity protection.

12.18. Cryptographic Binding

Since EAP-GPSK does not tunnel another EAP method, it does not implement cryptographic binding.

13. IANA Considerations

IANA has allocated a new EAP Type for EAP-GPSK (51).

IANA has created a new registry for ciphersuites, protected data types, failure codes, and op-codes. IANA has added the specified ciphersuites, protected data types, failure codes, and op-codes to these registries as defined below. Values defining ciphersuites (block-based or hash-based), protected data payloads, failure codes, and op-codes can be added or modified per IETF Review [RFC5226].

Figure 3 represents the initial contents of the "EAP-GPSK Ciphersuites" registry. The CSuite/Specifier field is 16 bits long. All other values are available via IANA registration. Each ciphersuite needs to provide processing rules and needs to specify how the following algorithms are instantiated: encryption, integrity, key derivation, and key length.

The following are the initial contents of the "EAP-GPSK Protected Data Payloads" registry:

- o 0x0000 : Reserved

The PData/Specifier field is 16 bits long, and all other values are available via IANA registration. Each extension needs to indicate whether confidentiality protection for transmission between the EAP peer and the EAP server is mandatory.

The following are the initial contents of the "EAP-GPSK Failure Codes" registry:

- o 0x00000000 : Reserved
- o 0x00000001 : PSK Not Found
- o 0x00000002 : Authentication Failure

- 0x00000003 : Authorization Failure

The Failure-Code field is 32 bits long, and all other values are available via IANA registration.

The following are the initial contents of the "EAP-GPSK OP Codes" registry:

- 0x00 : Reserved
- 0x01 : GPSK-1
- 0x02 : GPSK-2
- 0x03 : GPSK-3
- 0x04 : GPSK-4
- 0x05 : GPSK-Fail
- 0x06 : GPSK-Protected-Fail

The OP-Code field is 8 bits long, and all other values are available via IANA registration.

14. Contributors

This work is a joint effort of the EAP Method Update (EMU) design team of the EMU Working Group that was created to develop a mechanism based on strong shared secrets that meets RFC 3748 [RFC3748] and RFC 4017 [RFC4017] requirements. The design team members (in alphabetical order) were:

- Jari Arkko
- Mohamad Badra
- Uri Blumenthal
- Charles Clancy
- Lakshminath Dondeti
- David McGrew
- Joe Salowey
- Sharma Suman

o Hannes Tschofenig

o Jesse Walker

Finally, we would like to thank Thomas Otto for his reviews, feedback, and text contributions.

15. Acknowledgments

We would like to thank:

o Jouni Malinen and Bernard Aboba for their early comments on the document in June 2006. Jouni Malinen developed the first prototype implementation.

o Lakshminath Dondeti, David McGrew, Bernard Aboba, Michaela Vanderveen, and Ray Bell for their input to the ciphersuite discussions between July and August 2006.

o Lakshminath Dondeti for his detailed review (sent to the EMU mailing list on 12 July 2006).

o Based on a review requested from NIST, Quynh Dang suggested changes to the GKDF function (December 2006).

o Jouni Malinen and Victor Fajardo for their review in January 2007.

o Jouni Malinen for his suggestions regarding the examples and the key derivation function in February 2007.

o Bernard Aboba and Jouni Malinen for their review in February 2007.

o Vidya Narayanan for her review in March 2007.

o Pasi Eronen for his IESG review in March and July 2008.

o Dan Harkins for his review in June 2008.

o Joe Salowey, the EMU working group chair, provided a document review in April 2007. Jouni Malinen also reviewed the document during the same month.

o We would like to thank Paul Rowe, Arnab Roy, Prof. Andre Scedrov, and Prof. John C. Mitchell for their analysis of EAP-GPSK, for their input to the key derivation function, and for pointing us to a client-side DoS attack and to a downgrading attack. Based on their input, the key derivation function has been modified and the text in the security considerations section has been updated.

o Finally, we would like to thank our working group chair, Joe Salowey, for his support and for the time he spent discussing open issues with us.

16. References

16.1. Normative References

[AES] National Institute of Standards and Technology, "Specification for the Advanced Encryption Standard (AES)", Federal Information Processing Standards (FIPS) 197, November 2001.

[CBC] National Institute of Standards and Technology, "Recommendation for Block Cipher Modes of Encryption -- Methods and Techniques", Special Publication (SP) 800-38A, December 2001.

[CMAC] National Institute of Standards and Technology, "Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication", Special Publication (SP) 800-38B, May 2005.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, "Extensible Authentication Protocol (EAP)", RFC 3748, June 2004.

[RFC4282] Aboba, B., Beadles, M., Arkko, J., and P. Eronen, "The Network Access Identifier", RFC 4282, December 2005.

[RFC4634] Eastlake, D. and T. Hansen, "US Secure Hash Algorithms (SHA and HMAC-SHA)", RFC 4634, July 2006.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RFC5247] Aboba, B., Simon, D., and P. Eronen, "Extensible Authentication Protocol (EAP) Key Management Framework", RFC 5247, August 2008.

16.2. Informative References

[80211] "Information technology - Telecommunications and Information Exchange Between Systems - Local and Metropolitan Area Networks - Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications", IEEE Standard 802.11-2007, March 2007.

[ENTNUM] IANA, "SMI Network Management Private Enterprise Codes", Private Enterprise Numbers, <http://www.iana.org>.

[RFC4017] Stanley, D., Walker, J., and B. Aboba, "Extensible Authentication Protocol (EAP) Method Requirements for Wireless LANs", RFC 4017, March 2005.

[RFC4086] Eastlake, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, June 2005.

Authors' Addresses

T. Charles Clancy
DoD Laboratory for Telecommunications Sciences
8080 Greenmead Drive
College Park, MD 20740
USA

EMail: clancy@ltsnet.net

Hannes Tschofenig
Nokia Siemens Networks
Linnoitustie 6
Espoo 02600
Finland

EMail: Hannes.Tschofenig@gmx.net